Deterministic Regenerating Codes for Distributed Storage

Yunnan Wu*, Alexandros Dimakis†, Kannan Ramchandran†.
*Microsoft Research, Redmond. yunnanwu@microsoft.com
†Dept. of EECS, University of California, Berkeley. {adim,kannanr}@eecs.berkeley.edu

***Abstract*— It is well known that erasure coding can be used in storage systems to efficiently store data while protecting against failures. Conventionally, the design of erasure codes has focused on the tradeoff between redundancy and reliability; under this criterion, an Maximum Distance Separable (MDS) code is optimal. However, practical storage systems call for additional considerations. In particular, the codes must be properly maintained to recover from node failures. Previous work by Dimakis *et al.* studied the problem of properly maintaining erasure codes to reduce the incurred network bandwidth, established fundamental bounds on the minimum repair bandwidth for maintaining MDS codes, and showed that the repair bandwidth can be reduced further at the cost of higher storage. In this paper we present techniques for constructing codes that achieve the optimal tradeoffs between storage efficiency and repair bandwidth.**

I. Introduction

Many modern storage systems (e.g., peer-to-peer distributed storage [1], data center storage) achieve good reliability by storing the data redundantly in a large collection of unreliable storage nodes, such that when a small subset of nodes fail, the data can still be recovered from the surviving nodes. In this context, erasure coding naturally arises as an appealing technique. For instance, we can divide a file of size B into k pieces, each of size B/k, encode them into n coded pieces using an (n, k) maximum distance separable (MDS) code, and store them at n nodes. Then, the original file can be recovered from any set of k coded pieces. This is optimal in terms of the redundancy–reliability tradeoff because k pieces, each of size B/k, provide the minimum data for recovering the file, which is of size B.

However, in practical distributed storage systems, there are more considerations than achieving a good redundancy–reliability tradeoff. In particular, the codes must be properly repaired in the presence of node failures. Continuing with the (n, k) MDS code example, if a node holding a coded piece fails or leaves the system, in order to maintain the same level of reliability, we need to create a new encoded piece and store it at a new node (i.e., different from the surviving nodes holding the other $n - 1$ pieces). One straightforward way to do so is to let the new node download k encoded pieces from a subset of the surviving nodes, reconstruct the original file, and compute the needed new coded piece. In this process, the new node incurred a network traffic of $k \times B/k = B$. In practical storage systems, network bandwidth could be a critical resource. Hence an important consideration is to conserve the network bandwidth needed to repair the code in the presence of failures.

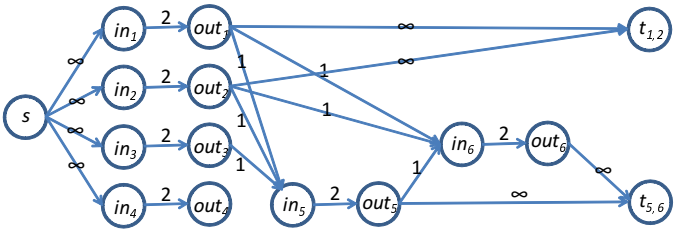

Fig. 1. The information flow graph.

In previous work [2], Dimakis *et al.* studied the problem of properly maintaining erasure codes to reduce the incurred network bandwidth. A key concept used in [2] is the *information flow graph*, which represents the evolution of information flow as nodes join and leave. Figure 1 gives an example information flow graph. In this graph, each storage node is represented by a pair of nodes, say in_i and out_i, connected by an edge whose capacity is the storage capacity of the node. There is a source node, s, which has the entire file. Suppose initially we store a file of 4 bytes at 4 nodes according to a $(4, 2)$ MDS code; this is represented in Figure 1 by the infinite capacity edges from the source s to the four storage nodes. Suppose storage node 4 fails next and we create a new storage node, node 5, which downloads 1 byte from each of the three surviving nodes and then stores 2 bytes; this is represented in Figure 1 by the unit-capacity edges out_1in_5, out_2in_5, and out_3in_5 that enter node 5. Similarly, the next part of the figure represents that a new storage node, node 6, is created, which downloads 1 byte each from nodes $1, 2, 5$ and stores 2 bytes. The rightmost part of Figure 1 contains two example *data collectors*, each corresponding to one request to reconstruct the original data from a subset of the nodes. The data collector $t_{1,2}$ has infinite capacity edges from nodes 1 and 2, modeling that it has access to nodes 1 and 2.

This information flow graph casts the original storage problem as a network communication problem where the source s multicasts the file to the set of all possible data collectors. By analyzing the connectivity in the information flow graph, we can derive fundamental performance bounds about codes. In particular, if the minimum cut between s and a data collector t_{ij} is less than the size of original file, then we can conclude that it is impossible for the data collector to reconstruct the original file by accessing storage nodes i and j, regardless

what code we use. Conversely, to ensure the minimum cut between s and a data collector t_{ij} is at least the size of the original file, we can derive necessary conditions regarding the storage capacities and the network bandwidth incurred (e.g., the 2's and 1's in Figure 1).

In addition to providing necessary conditions for all codes, the information flow graph can also imply the existence of codes under proper assumptions. For instance, consider an information flow graph G that enumerates all possible failure/repair patterns and all possible data collectors when the number of failures/repairs is bounded. If in G the min-cut between s and each data collector is at least the file size B, then from network coding theory (see, e.g., [3]), we know that there exists a linear code defined over a sufficiently large finite field $\mathbb{F}$ (whose size depends on the graph size) that ensures all data collectors can recover the original file. This implies that there exists a valid storage code achieving the necessary bound, which can tolerate a bounded number of failures/repairs.

By analyzing the connectivity of the information flow graph, Dimakis *et al.* [2] presented some important findings about storage codes and their maintenance. For instance, a notion called *Optimally Maintained MDS* (OMMDS) was introduced, which maintains an (n, k) MDS code with minimum network bandwidth when one node fails. It is proven there that if a new node can only connect to k nodes to download data for its new piece, then it has to download at least B, the size of the original file. However, a rather surprising result is that if this newcomer is allowed to connect to more than k nodes, then the network traffic can be reduced. In particular, if this newcomer is allowed to connect to all other $n-1$ surviving nodes, then it only needs to download $\frac{B}{(n-k)k}$ from each node, resulting in a total traffic of $\frac{(n-1)B}{(n-k)k}$. For example, in the (4,2) OMMDS scheme, a new node only needs to download a total of $0.75B$, as illustrated by Figure 1. Dimakis *et al.* [2] also proposed another scheme, which reduces the network bandwidth further than the OMMDS at the cost of increased storage redundancy. These schemes are called under the umbrella of "regenerating codes", because a node generally can upload to a newcomer a certain function of the bits it stored.

While the potential of regenerating codes has been demonstrated in [2], some important questions regarding their existence and construction remained unanswered. As we explained above, from existing network coding results, we can only conclude that there exists a valid storage code that can tolerate *a bounded number of failures/repairs*. However, it is undesirable to place a strict upper bound on the number of failures/repairs that can be tolerated. Would there exist a code that can work under an unbounded number of failures/repairs, assuming only the population of active nodes at any time is bounded? This is the question that this paper aims at answering.

The rest of the paper is organized as follows. In Section II, we review the flow analysis of regenerating codes based on [2] and present some extended results. The optimal tradeoffs between storage efficiency and repair bandwidth are characterized. In Section III, we present a method for constructing OMMDS codes that works under an unbounded number of failures/repairs, for the case a newcomer contacts all other $n-1$ nodes. In Section IV, we present a method for constructing non-MDS regenerating codes that achieve the optimal tradeoffs between storage efficiency and repair bandwidth, for the case a newcomer contacts all other $n-1$ nodes. Finally, a conclusion is given in Section V.

II. Flow Analysis of Regenerating Codes

In this section we analyze the connectivity in the information flow graph (as illustrated by Figure 1) to derive bounds for codes. The material in this section is based on [2] and extends [2]. The setup is as follows. There are always n active storage nodes. Each node can store α bits. Whenever a node fails, a newcomer downloads β bits each from d surviving nodes. We further restrict our attention to the symmetric setup where it is required that any k storage nodes can recover the original file. For each set of parameters (n, k, d, α, β), there is a family of finite or infinite information flow graphs, each of which corresponds to a particular evolution of node failures/repairs. We denote this family of directed acyclic graphs by $\mathcal{G}(n, k, d, \alpha, \beta)$.

Lemma 1: Consider any (potentially infinite) information flow graph G, formed by having n initial nodes that connect directly to the source and obtain α bits, while additional nodes join the graph by connecting to d existing nodes and obtaining β bits from each.[1] Any data collector t that connects to a k-subset of "out-nodes" (c.f. Figure 1) of G must satisfy:

$$\text{mincut}(s,t) \geq \sum_{i=0}^{\min\{d,k\}-1} \min\{(d-i)\beta, \alpha\}. \qquad (1)$$

Furthermore, there exists an information flow graph $G^* \in \mathcal{G}(n, k, d, \alpha, \beta)$ where this bound is matched with equality.

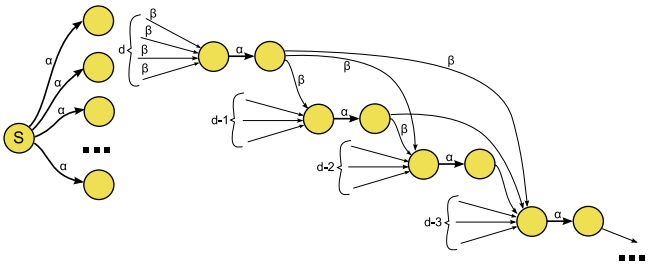

Fig. 2. G^* used in the proof of lemma 1

Proof: We first show that (1) must be satisfied for any G formed by adding d in-degree nodes as described above. Consider a data collector t that connects to a k-subset of "out-nodes", say $\{out_i : i \in I\}$. We want to show that any s–t cut in G has capacity at least $\sum_{i=0}^{\min\{d,k\}-1} \min\{(d-i)\beta, \alpha\}$. Since

[1]Note that this setup allows more graphs than those in $\mathcal{G}(n, k, d, \alpha, \beta)$. In a graph in $\mathcal{G}(n, k, d, \alpha, \beta)$, at any time there are n active storage nodes and a newcomer can only connect to the active nodes. In contrast, in a graph G described in this lemma, there is no notion of "active nodes" and a newcomer can connect to any d existing nodes.

the incoming edges of t all have infinite capacity, we only need to examine the cuts $(U, \overline{U})$ with $s \in U$, $out_i \in \overline{U}, \forall i \in I$. Let $\mathcal{C}$ denote the edges in the cut, i.e., the set of edges going from U to $\overline{U}$.

It is well known that each directed acyclic graph has a topological sorting (see, e.g., [4]), where a topological sorting (or acyclic ordering) is an ordering of its vertices such that the existence of an edge from v_i to v_j implies $i < j$. Let out_1 be the topologically first output node in $\overline{U}$. Consider two cases:

- If $in_1 \in U$, then the edge $in_1 out_1$ must be in $\mathcal{C}$.
- If $in_1 \in \overline{U}$, since in_1 has an in-degree of d and it is the topologically first node in $\overline{U}$, all the incoming edges of in_1 must be in $\mathcal{C}$.

Therefore, these edges related to out_1 will contribute a value of $\min\{d\beta, \alpha\}$ to the cut capacity.

Now consider out_2, the topologically second output node in $\overline{U}$. Similar to the above, we have two cases:

- If $in_2 \in U$, then the edge $in_2 out_2$ must be in $\mathcal{C}$.
- If $in_2 \in \overline{U}$, since at most one of the incoming edges of in_2 can be from out_1, $d-1$ incoming edges of in_1 must be in $\mathcal{C}$.

Following the same reasoning we find that for the i-th node ($i = 1, \ldots, \min\{d, k\}$) in the sorted set $\overline{U}$, either one edge of capacity α or $(d-(i-1))$ edges of capacity β each must be in $\mathcal{C}$. Equation (1) is exactly summing these contributions.

We now show that there exists an information flow graph G^* where the bound (1) is matched with equality. This graph is illustrated by Figure 2. In this graph, there are initially n nodes labeled from 1 to n. Consider k newcomers labeled as $n+1, \ldots, n+k$. The newcomer node $n+i$ connects to nodes $n+i-d, \ldots, n+i-1$. Consider a data collector t that connects to the last k nodes, i.e., nodes $n+1, \ldots, n+k$. Consider an s–t cut $(U, \overline{U})$ defined as follows. For each $i \in \{1, \ldots, k\}$, if $\alpha \leq (d-i)\beta$, then we include out_{n+i} in $\overline{U}$; otherwise, we include both out_{n+i} and in_{n+i} in $\overline{U}$. Then this cut $(U, \overline{U})$ achieves (1) with equality. ∎

From Lemma 1, we know that there exists a graph $G^* \in \mathcal{G}(n, k, d, \alpha, \beta)$ whose mincut is exactly $\sum_{i=0}^{\min\{d,k\}-1} \min\{(d-i)\beta, \alpha\}$. This implies that if we want to ensure recoverability while allowing a newcomer to connect to *any* set of d existing nodes, then the following is a necessary condition[2]

$$\sum_{i=0}^{\min\{d,k\}-1} \min\{(d-i)\beta, \alpha\} \geq B. \tag{2}$$

Furthermore, when this condition is satisfied, we know any graph in $\mathcal{G}(n, k, d, \alpha, \beta)$ will have enough flow from the source to each data collector. For this reason, we say

$$\mathsf{C} \triangleq \sum_{i=0}^{\min\{d,k\}-1} \min\{(d-i)\beta, \alpha\} \tag{3}$$

is the *capacity* for (n, k, d, α, β) regenerating codes (where each newcomer can access any arbitrary set of k nodes).

Note that if $d < k$, requiring any d storage nodes to have a flow of B will lead to the same condition (c.f. (2)) as requiring any k storage nodes to have a flow of B. Hence in such a case, we might as well set k as d. For this reason, in the following we assume $d \geq k$ without loss of generality.

We are interested in characterizing the achievable tradeoffs between the storage α and the repair bandwidth $d\beta$. To derive the optimal tradeoffs, we can fix the repair bandwidth and solve for the minimum α such that (2) is satisfied. For ease in derivation, we introduce a new parameter $\gamma \triangleq d\beta$; then the parameters $(n, k, d, \alpha, \gamma)$ can be used to characterize the system. With d and γ fixed, we minimize α; this can be cast as an optimization

$$\begin{aligned} \alpha^*(d, \gamma) \triangleq \min \quad & \alpha \\ \text{subject to:} \quad & \sum_{i=0}^{k-1} \min\left\{\left(1 - \frac{i}{d}\right)\gamma, \alpha\right\} \geq B. \end{aligned} \tag{4}$$

Lemma 2:

$$\alpha^*(d+1, \gamma) \leq \alpha^*(d, \gamma). \tag{5}$$

Hence a larger d always implies a better or equal storage–repair bandwidth tradeoff.

Proof: Show that $\alpha^*(d, \gamma)$ is always a feasible solution for the optimization for $\alpha^*(d+1, \gamma)$. ∎

The optimization (4) can be explicitly solved.

Lemma 3: For fixed d, the resulting function $\alpha^*(d, \gamma)$ is a piecewise linear function:

$$\alpha^*(d, \gamma) = \begin{cases} \frac{B}{k}, & \gamma \in [f(0), +\infty) \\ \frac{B - g(i)\gamma}{k-i}, & \gamma \in [f(i), f(i-1)), i = 1, \ldots, k-1 \end{cases} \tag{6}$$

where

$$f(i) \triangleq \frac{2Bd}{2ik - i^2 - i + 2k + 2kd - 2k^2}, \tag{7}$$

$$g(i) \triangleq \frac{(2d - 2k + i + 1)i}{2d}. \tag{8}$$

The minimum γ is:

$$\gamma_{\min} = f(k-1) = \frac{2Bd}{2kd - k^2 + k}. \tag{9}$$

It can be verified that the minimum storage point is achieved by the pair

$$(\alpha_0, \gamma_0) = \left(\frac{B}{k}, \frac{Bd}{k(d-k+1)}\right), \tag{10}$$

and the minimum repair bandwidth point is achieved by

$$(\alpha_1, \gamma_1) = \left(\frac{2Bd}{2kd - k^2 + k}, \frac{2Bd}{2kd - k^2 + k}\right). \tag{11}$$

For $k = 5$, $B = 1$, $d = 5, \ldots, 10$, the optimal tradoff curves are shown in Figure 3.

[2]This, however, does not rule out the possibility that the mincut is larger if a newcomer can choose the d existing nodes to connect to. We leave this as a future work.

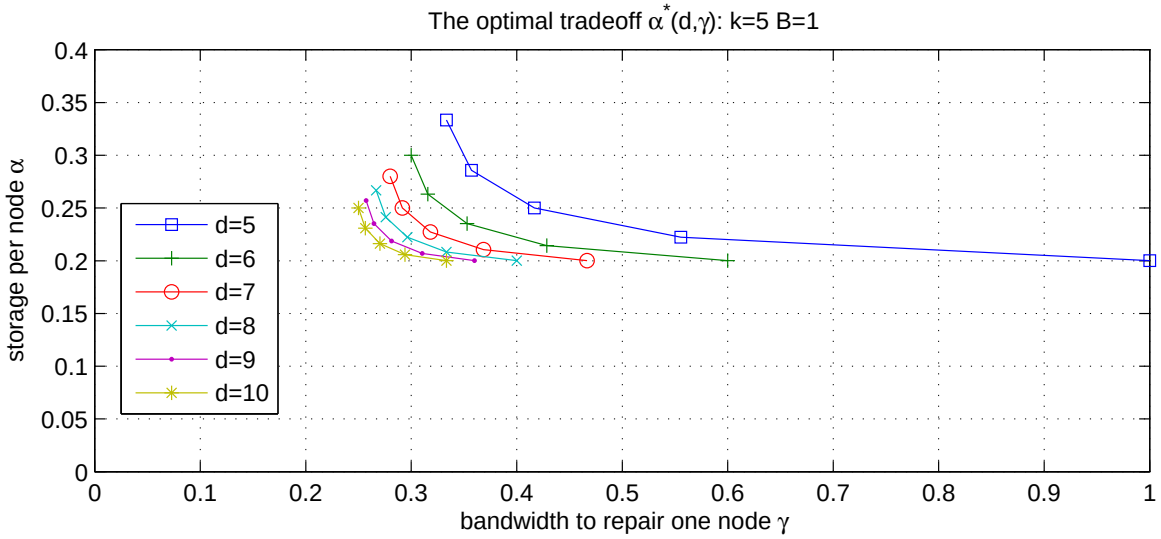

Fig. 3. The optimal tradoff curves between storage and repair bandwidth, for $d = 5, \ldots, 10$, $k = 5$, $B = 1$.

III. Deterministic Construction of OMMDS

In this section we present a deterministic construction of OMMDS via linear coding, for the case $d = n - 1$. Here, the parameters are $(n, k, d = n - 1, \alpha = \frac{B}{k}, \beta = \frac{B}{(n-k)k})$.

First, we explain the notations for linear coding. Suppose the original file is represented as a $(n-k)k \times W$ matrix $\boldsymbol{X}$ with entries defined in $\mathbb{F}$. Thus the original file can be viewed as $(n-k)k$ packets, each of size W. Each storage node thus can store $n-k$ packets. Suppose the i-th storage node stores $\boldsymbol{X}^T\boldsymbol{Q}_i$ where $\boldsymbol{Q}_i$ is a $(n-k)k \times (n-k)$ matrix. Hence the code is completely specified by $\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_n$; we will view each matrix $\boldsymbol{Q}_i$ as $n-k$ code vectors, each of length $(n-k)k$. We want to maintain that at any time any set of k nodes enable the original file to be reconstructed; equivalently, we want to maintain the following property.

Property 1 (MDS Property): For any set of k nodes, say $s_1, \ldots, s_k$, the span of the $(n-k)k$ code vectors in $\{\boldsymbol{Q}_{s_1}, \ldots, \boldsymbol{Q}_{s_k}\}$ has full rank.

The following lemma is a linear algebra property.

Lemma 4: Consider a code $\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_n$ that satisfies the MDS Property. For any permutation of the n nodes, say $s_1, \ldots, s_n$, it is possible to select $n-k$ vectors from each of $s_1, \ldots, s_{k-1}$ and 1 vector from each of $s_k, \ldots, s_{n-1}$ such that the resulting set of $(n-k)(k-1) + (n-k) = (n-k)k$ vectors has full rank.

Proof: Let S initially be the set of $(k-1)(n-k)$ linearly independent vectors consisting of the column vectors of $\boldsymbol{Q}_{s_1}, \ldots, \boldsymbol{Q}_{s_{k-1}}$. For each node in $\{s_k, \ldots, s_{n-1}\}$, we examine its $n-k$ vectors one by one and find one that is linearly independent from the current S; if so, we add this vector, denoted by $\boldsymbol{q}_i$, to S.

We now prove by a contradiction argument that we can always find one such vector for $i \in \{s_k, \ldots, s_{n-1}\}$. If we cannot find such a qualifying vector for i, then $\{\boldsymbol{Q}_{s_1}, \ldots, \boldsymbol{Q}_{s_{k-1}}, \boldsymbol{Q}_i\} \subseteq \text{span}(S)$. From the inductive hypothesis, $\{\boldsymbol{Q}_{s_1}, \ldots, \boldsymbol{Q}_{s_{k-1}}, \boldsymbol{Q}_i\}$ has full rank; thus this implies that S has rank $(n-k)k$. However, since we failed in the vector addition process, S only contains less than $(n-k)k$ vectors. This leads to a contradition. Hence the claim is established.■

Lemma 5 (Schwartz–Zippel Thorem (see, e.g., [5])): Let $Q(x_1, \ldots, x_n) \in \mathbb{F}[x_1, \ldots, x_n]$ be a multivariate polynomial of total degree d_0 (the total degree is the maximum degree of the additive terms and the degree of a term is the sum of exponents of the variables). Fix any finite set $\mathbb{S} \subseteq \mathbb{F}$, and let $r_1, \ldots, r_n$ be chosen independently and uniformly at random from $\mathbb{S}$. Then if $Q(x_1, \ldots, x_n)$ is not equal to a zero polynomial,

$$\Pr[Q(r_1, \ldots, r_n) = 0] \leq \frac{d_0}{|\mathbb{S}|}. \tag{12}$$

Theorem 1: Given $(n, k, d = n-1, \alpha = \frac{B}{k}, \beta = \frac{B}{(n-k)k})$, for any finite field $\mathbb{F}$ of size greater than

$$d_0 = \binom{n}{k} (n-k)k, \tag{13}$$

there exists a linear OMMDS coding scheme defined in $\mathbb{F}$, such that at any moment the MDS Property is satisfied, regardless of how many failures/repairs happened.

Proof: The proof is by induction. We maintain the MDS Property as an invariant. Initially we choose $[\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_n]$ such that the MDS Property is satisfied. This is equivalent to picking $(n-k)n$ coefficients such that:

$$\prod_{\{s_1, \ldots, s_k\} \subseteq \{1, \ldots, n\}} \det\left([\boldsymbol{Q}_{s_1}, \ldots, \boldsymbol{Q}_{s_k}]\right) \neq 0. \tag{14}$$

It will become clear later in this proof that this is possible as long as the field satisfies $|\mathbb{F}| > d_0$.

Suppose the MDS Property is currently satisfied. Suppose node 1 fails next and a newcomer arrives. This newcomer needs to download one packet from each node, compute $(n-k)$ packets from the downloaded $(n-1)$ packets, and store them. Since we use linear codes, let the coding matrix at the newcomer be:

$$\boldsymbol{Q}_1' = [\boldsymbol{Q}_2\boldsymbol{b}_2, \ldots, \boldsymbol{Q}_n\boldsymbol{b}_n]\boldsymbol{Z}. \tag{15}$$

Here $\boldsymbol{b}_i$ is of size $(n-k)\times 1$ and $\boldsymbol{Z}$ is of size $(n-1)\times(n-k)$; $\boldsymbol{X}^T\boldsymbol{Q}_i\boldsymbol{b}_i$ is the packet the newcomer downloaded from node i; $\boldsymbol{Z}$ is the linear transformation applied to the $n-1$ received packets.

Here $\{\boldsymbol{b}_2, \ldots, \boldsymbol{b}_n\}$ and $\boldsymbol{Z}$ are the free parameters we can tune. Collectively, we use $\boldsymbol{p} \in \mathbb{F}^t$ to denote these free parameters, where $t \triangleq 2(n-k)(n-1)$. We will show that there exists an assignment of these free parameters such that $[\boldsymbol{Q}_1', \boldsymbol{Q}_2, \ldots, \boldsymbol{Q}_n]$ continues to satisfy the MDS Property.

Since $[\boldsymbol{Q}_1, \boldsymbol{Q}_2, \ldots, \boldsymbol{Q}_n]$ satisfies the MDS Property, we only need to ensure that for every subset of $k-1$ existing nodes, say $\{s_1, \ldots, s_{k-1}\} \subseteq \{2, \ldots, n\}$, the span of the $k(n-k)$ code vectors in $\{\boldsymbol{Q}_1', \boldsymbol{Q}_{s_1}, \ldots, \boldsymbol{Q}_{s_{k-1}}\}$ has full rank. In other words, we want to show there exists $\boldsymbol{p} \in \mathbb{F}^t$ such that

$$\prod_{\{s_1,\ldots,s_{k-1}\}\subseteq\{2,\ldots,n\}} \det\left([\boldsymbol{Q}_1', \boldsymbol{Q}_{s_1}, \ldots, \boldsymbol{Q}_{s_{k-1}}]\right) \neq 0. \tag{16}$$

To do so, we will interpret the left hand side of (16) in two ways: (i) as a number in $\mathbb{F}$ (ii) as a polynomial in terms of the $2(n-k)(n-1)$ variables in $\boldsymbol{p}$, with total degree at most

$$\begin{pmatrix} n-1 \\ k-1 \end{pmatrix}(n-k)2, \tag{17}$$

which is less than or equal to d_0 if $n > 1$. (The degree bound follows from that each entry of $\boldsymbol{Q}_1'$ is at most of degree 2, and then the determinant of an $n \times n$ matrix $\boldsymbol{M}$ is $\det(\boldsymbol{M}) = \sum_{\pi\in\mathbb{S}_n} \text{sgn}(\pi) \prod_{i=1}^n M_{i,\pi(i)}$, where $\mathbb{S}_n$ is the symmetric group of permutations of size n, and $\text{sgn}(\pi) = (-1)^t$ where t is the number of pairwise element exchanges required to transform the identity permutation into π.) We will establish the following one by one:

(i) For each $\{s_1, \ldots, s_{k-1}\} \subseteq \{2, \ldots, n\}$, $\det\left([\boldsymbol{Q}_1', \boldsymbol{Q}_{s_1}, \ldots, \boldsymbol{Q}_{s_{k-1}}]\right)$ is a nonzero number for a certain assignment of $\boldsymbol{p} \in \mathbb{F}^t$.
(ii) $\det\left([\boldsymbol{Q}_1', \boldsymbol{Q}_{s_1}, \ldots, \boldsymbol{Q}_{s_{k-1}}]\right)$ is a nonzero polynomial.
(iii) The left hand side of (16) as a polynomial is nonzero.
(iv) If $|\mathbb{F}| > d_0$, then there exists an assignment of $\boldsymbol{p} \in \mathbb{F}^t$ such that (16) holds, according to the Schwartz–Zippel Theorem (Lemma 5).

We now establish step (i). For each $\{s_1, \ldots, s_{k-1}\} \subseteq \{2, \ldots, n\}$, let $\{s_k, \ldots, s_{n-1}\} = \{2, \ldots, n\} - \{s_1, \ldots, s_{k-1}\}$. From Lemma 4, we can select one vector, say $\boldsymbol{q}_i$, from each node i in $\{s_k, \ldots, s_{n-1}\}$ such that $\{\boldsymbol{Q}_{s_1}, \ldots, \boldsymbol{Q}_{s_{k-1}}, \boldsymbol{q}_{s_k}, \ldots, \boldsymbol{q}_{s_{n-1}}\}$ has full rank. This means that we can set $\{\boldsymbol{b}_i\}$ and $\boldsymbol{Z}$ so that the newcomer stores the $n-k$ vectors $\boldsymbol{q}_{s_k}, \ldots, \boldsymbol{q}_{s_{n-1}}$ as $\boldsymbol{Q}_1'$; this satisfies that $\det\left([\boldsymbol{Q}_1', \boldsymbol{Q}_{s_1}, \ldots, \boldsymbol{Q}_{s_{k-1}}]\right)$ is a nonzero number. This establishes (i). The remaining steps (ii)(iii)(iv) are straightforward.

Via a similar argument, it can be shown that there exists an initial code assignment satisfying the MDS Property if $|\mathbb{F}| > d_0$. Hence the claim is established for $|\mathbb{F}| > d_0$. ■

In the above we have shown the existence of a linear OMMDS code for any finite field of size at least d_0. To construct the code explicitly, we can use a large enough finite field so that the probability of failure (12) is small and hence the expected complexity to find a valid code is low.

Alternatively, we can use a non-randomized algorithm to find the codes, similar to the algorithm used in Jaggi *et al.* [6]. We give an outline as follows. For the initialization stage, we can use existing methods for constructing MDS, e.g., by directly applying Jaggi *et al.*'s algorithm [6]. For the repair stage, the code update problem can be converted into a virtual multicast problem. Lemma 4 can be used to show that we can execute Jaggi *et al.*'s algorithm from the middle, where some coefficients have already been determined and some other coefficients are to be determined. Then Jaggi *et al.*'s algorithm can be applied to deterministically decide the coefficients.

IV. Deterministic Construction of Non-MDS Regenerating Codes for $d = n-1$

In this section we discuss the deterministic construction of non-MDS regenerating codes achieving points on the optimal tradeoff curve for $d = n-1$. Without essential loss of generality, we assume that α and β are both integers. Suppose the units for α, β, B are measured in packets and the file is represented as

$$B = \sum_{i=0}^{k-1} \min\{(d-i)\beta, \alpha\} \tag{18}$$

packets. Each storage node can store α packets. Suppose the coding matrix for the i-th node is $\boldsymbol{Q}_i$, which is an $B\times\alpha$ matrix and can be viewed as α code vectors, each of length B. Each time a newcomer arrives, it downloads β packets from each of the surviving nodes. We want to maintain that at any time any k nodes enable the original file to be reconstructed.

Our proof hinges upon the following notion.

Definition 1 (Rank Test): A linear code $\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_n$ defined in $\mathbb{F}$ is said to pass the rank test by a test vector $\boldsymbol{h} = (h_1, \ldots, h_n)$ (where $h_i \in \{0, \ldots, \alpha\}$) if $[\boldsymbol{Q}_1\boldsymbol{E}_{h_1}, \ldots, \boldsymbol{Q}_n\boldsymbol{E}_{h_n}]$ has full rank B, where $\boldsymbol{E}_{h_i}$ is a selection matrix so that $\boldsymbol{Q}_i\boldsymbol{E}_{h_i}$ is the first h_i columns of $\boldsymbol{Q}_i$. This full rank condition is equivalent to that there exists a matrix $\boldsymbol{D}$ of size $\sum_{i=1}^n h_i \times B$ such that

$$\det\left([\boldsymbol{Q}_1\boldsymbol{E}_{h_1}, \ldots, \boldsymbol{Q}_n\boldsymbol{E}_{h_n}]\boldsymbol{D}\right) \neq 0. \tag{19}$$

To ease the explanation, we sometimes simply say "the code passes $\boldsymbol{h}$".

We also define a weaker notion. A code passes a rank test by a test vector $\boldsymbol{h}$ *weakly* if we can generate h_i vectors by taking linear combinations of the columns of each $\boldsymbol{Q}_i$ such that the resulting set has rank B.

Using the notion of rank tests, our objective can be equivalently stated as ensuring that at any time the code $\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_n$ passes the rank tests by the following set of vectors:

$$\mathcal{H}_0 \triangleq \left\{ \boldsymbol{h} \,\middle|\, \boldsymbol{h} \text{ is a permutation of } \boldsymbol{h}^{(0)} \right\} \tag{20}$$

$$\boldsymbol{h}^{(0)} \triangleq (\underbrace{\alpha, \ldots, \alpha}_{k}, \underbrace{0, \ldots, 0}_{n-k}) \tag{21}$$

Before going to the proof, let us recapitulate how we did the OMMDS proof, using the notion of rank tests. In the OMMDS case, the proof is done by induction: Given the old code passes the rank tests by all vectors in $\mathcal{H}_0$, we first showed via linear algebra (Lemma 4) that this implies that the old code passes the rank tests weakly by all vectors in $\mathcal{H}_1$, where $\mathcal{H}_1$ is defined as:

$$\mathcal{H}_1 \triangleq \left\{ \boldsymbol{h} \,\middle|\, \boldsymbol{h} \text{ is a permutation of } \boldsymbol{h}^{(1)} \right\} \tag{22}$$

$$\boldsymbol{h}^{(1)} \triangleq (\underbrace{\alpha, \ldots, \alpha}_{k-1}, \underbrace{\beta, \ldots, \beta}_{n-k}, 0) \tag{23}$$

This implies it is possible to find coefficients such that the updated code, $\boldsymbol{Q}_1', \boldsymbol{Q}_2, \ldots, \boldsymbol{Q}_n$, passes each test by a vector $\boldsymbol{h} \in \mathcal{H}_0$. Then we used the technique of switching back and forth between two interpretations of the product of determinants (16) (originally introduced by Koetter and Medard [7]) to show there exists one set of coefficients such that the updated code passes all tests in $\mathcal{H}_0$ simultaneously. This is how we proved the induction "Old code passes $\mathcal{H}_0$" implies "Updated code can pass $\mathcal{H}_0$". In the non-MDS case, such an induction is no longer possible. Instead, we conduct induction using a larger set of test vectors $\mathcal{H} \supset \mathcal{H}_0$. The induction is: "Old code passes $\mathcal{H}$" implies "Updated code can pass $\mathcal{H}$". Specifically, to do the induction, we use a set of test vectors $\mathcal{H}$ that satisfies: For any $\boldsymbol{h} \in \mathcal{H}$, there exists a vector $\boldsymbol{h}' \in \mathcal{H}$ such that "Old code passes $\boldsymbol{h}'$" implies "Updated code can pass $\boldsymbol{h}$".

We now explain the basic idea of our proof technique with an example in Figure 4, where $n = 4$, $k = 2$, $d = 3$, $B = 5$, $\alpha = 3$, $\beta = 1$. The old code is formed by $\boldsymbol{Q}_1, \boldsymbol{Q}_2, \boldsymbol{Q}_3, \boldsymbol{Q}_4$, which are stored at the four nodes in the box; the updated code is formed by $\boldsymbol{Q}_1, \boldsymbol{Q}_2, \boldsymbol{Q}_3, \boldsymbol{Q}_5$, where $\boldsymbol{Q}_5$ is the coding matrix at the newcomer node 5. Define $\mathcal{H}$ as the set of permutations of the following three vectors:

$$\boldsymbol{h}^{(0)} = (3, 3, 0, 0), \tag{24}$$

$$\boldsymbol{h}^{(1)} = (3, 1, 1, 0), \tag{25}$$

$$\boldsymbol{h}^{(2)} = (2, 2, 1, 0). \tag{26}$$

Suppose the new code is challenged by the test vector $\boldsymbol{h} = (h_1 = 0, h_2 = 0, h_3 = 3, h_5 = 3)$. Then to show that the new code can pass the test $\boldsymbol{h} = (h_1 = 0, h_2 = 0, h_3 = 3, h_5 = 3)$, we use the fact the old code passes the test $\boldsymbol{h}' = (h_1 = 1, h_2 = 1, h_3 = 3, h_4 = 0)$. Specifically, let $\boldsymbol{q}_{i,j}$ stand for the j-th code vector of node i; then the set $\{\boldsymbol{q}_{1,1}, \boldsymbol{q}_{2,1}, \boldsymbol{q}_{3,1}, \boldsymbol{q}_{3,2}, \boldsymbol{q}_{3,3}\}$ has rank 5. If node 5 stores $\boldsymbol{q}_{1,1}$ as $\boldsymbol{q}_{5,1}$ and $\boldsymbol{q}_{2,1}$ as $\boldsymbol{q}_{5,2}$, then $\{\boldsymbol{q}_{5,1}, \boldsymbol{q}_{5,2}, \boldsymbol{q}_{5,3}, \boldsymbol{q}_{3,1}, \boldsymbol{q}_{3,2}, \boldsymbol{q}_{3,3}\}$ has rank 5; thus the updated code can pass the test by $\boldsymbol{h}$. The other tests can be checked in a similar manner.

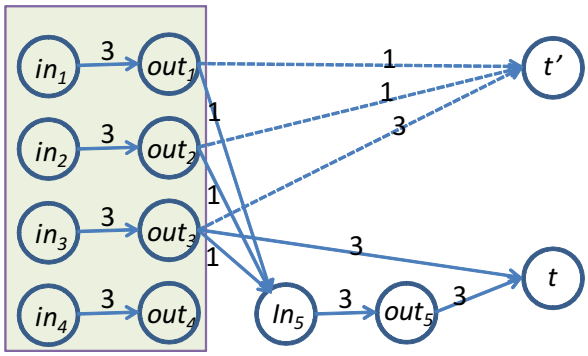

Fig. 4. Illustration of the proof technique.

The above argument can be interpreted as an algebraic "path-weaving" argument. As illustrated in Figure 4, the test vector $\boldsymbol{h}$ corresponds to the data collector t who wants to reconstruct the file by accessing nodes 3 and 5; the test $\boldsymbol{h}'$ corresponds to the data collector t' who wants to reconstruct the file by downloading 1 packet from node 1, 1 packet from node 2, and 3 packets from node 3. If there are 5 unit-rate parallel paths carrying information to t', then there are 5 paths carrying information to t because we can simply redirect the path involving node 1 and the path involving node 2 via node 5 to node t.

Having explained the basic idea, we now present the results more formally. Note that the following theorem holds for general (n, k, d, α, β) where α and β are integers.

Theorem 2: Assume we have a set of test vectors $\mathcal{H}$ that satisfies the following properties:

- $\mathcal{H} \supset \mathcal{H}_0$;
- $\forall \boldsymbol{h} \in \mathcal{H}$, each entry $h_i \in \{0, \ldots, \alpha\}$;
- $\forall \boldsymbol{h} \in \mathcal{H}$, $w(\boldsymbol{h}) \triangleq \sum_{i=1}^{n} h_i \geq B$;
- For any $\boldsymbol{h} \in \mathcal{H}$ and any permutation of $1, \ldots, n$, $s_1, \ldots, s_n$, there exists a vector $\boldsymbol{h}' \in \mathcal{H}$ such that

$$h'_{s_1} = 0, \tag{27}$$

$$h_{s_1} \geq \sum_{i=2}^{d+1} (h'_{s_i} - h_{s_i}) \tag{28}$$

$$h'_{s_i} \in [h_{s_i}, h_{s_i} + \beta], \text{ for } i = 2, \ldots, d+1, \tag{29}$$

$$h'_{s_i} = h_{s_i}, \text{ for } i = d+2, \ldots, n. \tag{30}$$

Let $\mathbb{F}$ be a finite field whose size is greater than

$$d_1 \triangleq \max \left\{ \binom{n\alpha}{B} B, \; 3|\mathcal{H}|B \right\} \tag{31}$$

Then, there exists a linear scheme defined in $\mathbb{F}$ that achieves the capacity for (n, k, d, α, β).

Proof: The proof is again based on induction. We again use the technique of switching back and forth between two interpretations of the product of determinants (16) (originally introduced by Koetter and Medard [7]).

The invariant we maintain in the induction is that at any time the code passes $\mathcal{H}$. This can be stated as the following condition:

$$\prod_{\boldsymbol{h}\in\mathcal{H}:\ h_1>0} \det\left([\boldsymbol{Q}_1\boldsymbol{E}_{h_1},\ldots,\boldsymbol{Q}_n\boldsymbol{E}_{h_n}]\boldsymbol{D}_{\boldsymbol{h}}\right)\neq 0, \qquad (32)$$

for some $\{\boldsymbol{D}_{\boldsymbol{h}}\}$.

Initially, let $\boldsymbol{G}$ (of size $B\times n\alpha$) be the generator matrix of an $(n\alpha, B)$ MDS code defined in $\mathbb{F}$. It can be shown via techniques similar to the proof of Theorem 1 that this is possible. Then we let $[\boldsymbol{Q}_1,\ldots,\boldsymbol{Q}_n]$ be $\boldsymbol{G}$; i.e., $\boldsymbol{Q}_1$ is the first α columns of $\boldsymbol{G}$, and so on. From the $(n\alpha, B)$ MDS property, we know that the initial code passes $\mathcal{H}$.

Suppose node s_1 fails next and a newcomer arrives. This newcomer needs to download β packets from each of d nodes (say, $s_2,\ldots,s_{d+1}$), compute α packets from the downloaded $d\beta$ packets, and store them. Since we use linear codes, let the coding matrix at the newcomer be:

$$\boldsymbol{Q}'_{s_1}=[\boldsymbol{Q}_{s_2}\boldsymbol{B}_{s_2},\ldots,\boldsymbol{Q}_{s_{d+1}}\boldsymbol{B}_{s_{d+1}}]\boldsymbol{Z}. \qquad (33)$$

Here $s_1,\ldots,s_n$ constitutes a permutation of $1,\ldots,n$, $\boldsymbol{B}_i$ is of size $\alpha\times\beta$ and $\boldsymbol{Z}$ is of size $d\beta\times\alpha$.

We now show for any $\boldsymbol{h}\in\mathcal{H}$, the updated code can pass the test of $\boldsymbol{h}$. Let $\boldsymbol{h}'\in\mathcal{H}$ be a vector as assumed in the theorem statement. From the inductive hypothesis, the set formed by the first h'_i vectors from each $\boldsymbol{Q}_i$ has rank B. Now let the newcomer download $h'_{s_i}-h_{s_i}$ coding vectors from s_i, for $i=2,\ldots,d+1$. Since $\alpha\geq h_{s_1}\geq\sum_{i=2}^{d+1}(h'_{s_i}-h_{s_i})$, these packets (or corresponding coding vectors) can be stored at node s_1 and accessed when taking the test by $\boldsymbol{h}$. These vectors, together with the first h_{s_i} vectors from each s_i for $i\in\{2,\ldots,n\}$, have rank B. This shows that there exists an assignment of the parameters such that the updated code passes the test of $\boldsymbol{h}$. Thus, each determinant term in the product (32) is a nonzero number for certain assignment of the coefficients $\boldsymbol{B}_2,\ldots,\boldsymbol{B}_n,\boldsymbol{Z},\boldsymbol{D}_{\boldsymbol{h}}$. Hence each term in the product is a nonzero polynomial. Hence the product is a nonzero polynomial. The total degree of this polynomial can be shown to be at most d_1. Finally, we can apply the Schwartz–Zippel Theorem (Lemma 5) to establish the proof. ∎

It remains to specify the set of test vectors $\mathcal{H}$ assumed in the theorem. Theorem 2 holds for general parameters (n,k,d,α,β). In this paper we show how to find $\mathcal{H}$ for the case $d=n-1$; the general case is left as a future investigation. Since $d=n-1$, $B=\sum_{i=0}^{k-1}\min\{(n-1-i)\beta,\alpha\}$. We represent α as follows:

$$\alpha=(n-k+m-1)\beta+r \qquad (34)$$

where $r\in[0,\beta)$ is the remainder. If $\alpha<(n-k)\beta$, then it follows that $\alpha=B/k$; in this case, we can use the technique of Section III to design an OMMDS with a smaller β. If $\alpha>(n-1)\beta$, then we can design a code with α set to a smaller value $(n-1)\beta$. Therefore, we will focus on the cases where $m=1,\ldots,k$.

We now give one specification of the set of test vectors $\mathcal{H}$. For $j=0,\ldots,m$, define

$$\boldsymbol{h}^{(j)}\triangleq(\underbrace{\alpha,\ldots,\alpha}_{k-j},\underbrace{j\beta,\ldots,j\beta}_{n-k},\underbrace{(j-1)\beta,\ldots,2\beta,\beta,0}_{j}) \qquad (35)$$

For $j=m+1,\ldots,k$, define

$$\boldsymbol{h}^{(j)}\triangleq\mathsf{sort}_d(\boldsymbol{h}^{(j-1)}+\boldsymbol{u}), \qquad (36)$$

where

$$\boldsymbol{u}\triangleq(-\alpha,\underbrace{0,\ldots,0}_{k-m-1},r,\underbrace{\beta,\ldots,\beta}_{n-k+m-1}), \qquad (37)$$

and $\mathsf{sort}_d(\boldsymbol{x})$ is the vector obtained by sorting the elements of $\boldsymbol{x}$ in decreasing order.

Let $\boldsymbol{h}^{(m')}$ be the minimum indexed vector in $\boldsymbol{h}^{(0)},\ldots,\boldsymbol{h}^{(k)}$ that is β-gradually-decreasing (see Definition 2); if no vector in $\boldsymbol{h}^{(0)},\ldots,\boldsymbol{h}^{(k)}$ is β-gradually-decreasing, let $m'=k$. Then we define $\mathcal{H}$ as the set of all permutations of $\boldsymbol{h}_0,\ldots,\boldsymbol{h}_{m'}$. For an example, see the discussion before Theorem 2.

Definition 2 (β-gradually-decreasing vectors): A vector $(v_1,\ldots,v_n)$ is said to be "β-gradually-decreasing" if $v_1\geq v_2\geq\ldots\geq v_n$ and the difference between any two consecutive elements is at most β. A vector $(u_1,\ldots,u_n)$ is said to be "β-gradually-increasing" if $u_1\leq u_2\leq\ldots\leq u_n$ and the difference between any two consecutive elements is at most β.

Lemma 6: Let $(v_1,\ldots,v_n)$ be a β-gradually-decreasing vector and $(u_1,\ldots,u_n)$ be a β-gradually-increasing vector. Then $\mathsf{sort}_d(v_1+u_1,\ldots,v_n+u_n)$ is a β-gradually-decreasing vector.

Proof: Note that for any i, $v_i\in[v_{i+1},v_{i+1}+\beta]$ and $u_i\in[u_{i+1}-\beta,u_{i+1}]$. Thus $|(v_i+u_i)-(v_{i+1}+u_{i+1})|\leq\beta$. We now show by induction that $\mathsf{sort}_d(v_1+u_1,\ldots,v_i+u_i)$, for $i=1,\ldots,n$, is β-gradually-decreasing. The case $i=1$ is trivial. Assume $\mathsf{sort}_d(v_1+u_1,\ldots,v_i+u_i)$ is β-gradually-decreasing. Now we insert the value $v_{i+1}+u_{i+1}$ in the sorted sequence. Since $|(v_i+u_i)-(v_{i+1}+u_{i+1})|\leq\beta$, $\mathsf{sort}_d(v_1+u_1,\ldots,v_{i+1}+u_{i+1})$ remains to be β-gradually-decreasing. Hence the inductive claim is established. ∎

Theorem 3: The set of test vectors $\mathcal{H}$ specified above satisfies:

1. $\mathcal{H}\supset\mathcal{H}_0$;
2. $\forall\boldsymbol{h}\in\mathcal{H}$, $w(\boldsymbol{h})\triangleq\sum_{i=1}^{n}h_i\geq B$;
3. $\forall\boldsymbol{h}\in\mathcal{H}$, each entry $h_i\in\{0,\ldots,\alpha\}$;
4. For $j=0,\ldots,m'$, $\boldsymbol{h}^{(j)}$ consists of $(k-j)$ α's followed by a β-gradually-decreasing vector ending in 0.
5. For any $\boldsymbol{h}\in\mathcal{H}$ and any permutation of $1,\ldots,n$,

$s_1, \ldots, s_n$, there exists a vector $\boldsymbol{h}' \in \mathcal{H}$ such that

$$\begin{aligned} & h'_{s_1} = 0, \\ & h_{s_1} \geq \sum_{i=2}^{n} (h'_{s_i} - h_{s_i}) \\ & h'_{s_i} \in [h_{s_i}, h_{s_i} + \beta], \text{ for } i = 2, \ldots, n. \end{aligned} \tag{38}$$

Proof: Property 1 is trivial because $\mathcal{H}$ includes the permutations of $\boldsymbol{h}^{(0)}$. For Property 2, it can be checked that

$$w(\boldsymbol{h}^{(0)}) \geq w(\boldsymbol{h}^{(1)}) \geq \ldots \geq w(\boldsymbol{h}^{(m)}) = \ldots = w(\boldsymbol{h}^{(k)}) = B.$$

We now verify Properties 3 and 4 for $\boldsymbol{h}^{(0)}, \ldots, \boldsymbol{h}^{(m)}$. Since we assumed that $k \leq d$ and $d = n - 1$, $m\beta \leq (n - k + m - 1)\beta + r = \alpha$. Thus Property 3 is satisfied for $\boldsymbol{h}^{(0)}, \ldots, \boldsymbol{h}^{(m)}$. It is clear that Property 4 holds for $\boldsymbol{h}^{(0)}, \ldots, \boldsymbol{h}^{(m)}$.

We now prove by induction that Properties 3 and 4 hold for each $\boldsymbol{h}^{(j)}$ for $j = m, \ldots, m'$. The case $j = m$ has already been established. Suppose each entry of $\boldsymbol{h}^{(j)}$ is in $\{0, \ldots, \alpha\}$ and $\boldsymbol{h}^{(j)}$ consists of $(k - j)$ α's followed by a β-gradually-decreasing vector ending in 0. Let

$$\boldsymbol{h}^{(j)} = (\underbrace{\alpha, \ldots, \alpha}_{k-j}, \underbrace{z_{n-k+j}, \ldots, z_1 = 0}_{n-k+j}). \tag{39}$$

Since

$$\boldsymbol{u} = (-\alpha, \underbrace{0, \ldots, 0}_{k-m-1}, r, \underbrace{\beta, \ldots, \beta}_{n-k+m-1})$$

we know

$$\begin{aligned} \boldsymbol{h}^{(j)} + \boldsymbol{u} = (0, \underbrace{\alpha, \ldots, \alpha}_{k-j-1}, & z_{n-k+j}, \ldots, z_{n-k+m+1}, \\ & z_{n-k+m} + r, z_{n-k+m-1} + \beta, \ldots, \beta). \end{aligned} \tag{40}$$

Since $(z_{n-k+j}, \ldots, z_1)$ is a β-gradually-decreasing vector ending in 0, $z_{n-k+m} \leq (n - k + m - 1)\beta$ and $z_{n-k+m-1} \leq (n-k+m-2)\beta$. Thus $z_{n-k+m} + r \leq \alpha$ and $z_{n-k+m-1} + \beta \leq \alpha$. This establishes that each entry of $\boldsymbol{h}^{(j+1)}$ is in $\{0, \ldots, \alpha\}$.

From (40), and noting the fact $\boldsymbol{u}$ is β-gradually-increasing and $(z_{n-k+j}, \ldots, z_1)$ is β-gradually-decreasing, we conclude (by Lemma 6) that $\boldsymbol{h}^{(j+1)}$ consists of $(k-j-1)$ α's followed by a β-gradually-decreasing vector ending in 0.

By now we have established that $\mathcal{H}$ satisfies Properties 1,2,3,4. We now prove that $\mathcal{H}$ satisfies Property 5. Since $\mathcal{H}$ is defined by the permutations of the vectors $\boldsymbol{h}^{(0)}, \ldots, \boldsymbol{h}^{(m')}$, we only need to show that for any $\boldsymbol{h} \in \{\boldsymbol{h}^{(0)}, \ldots, \boldsymbol{h}^{(m')}\}$, there exists a vector $\boldsymbol{h}'$ satisfying (38). Consider $\boldsymbol{h} = \boldsymbol{h}^{(j)}$ and an arbitrary permutation of $1, \ldots, n$, say $s_1, \ldots, s_n$. Consider three cases.

Case 1: If $j = m'$ or $s_1 > k - j$; in either case the subvector $(h_{s_1}, \ldots, h_n)$ is β-gradually-decreasing. To satisfy (38), we choose

$$\boldsymbol{h}' = (h_1, \ldots, h_{s_1 - 1}, h_n = 0, h_{s_1}, \ldots, h_{n-1}), \tag{41}$$

which is a permutation of $\boldsymbol{h}$ and hence is in $\mathcal{H}$.

Case 2: If $j \in \{0, \ldots, m-1\}$ and $s_1 \leq k - j$, then

$$\boldsymbol{h} = \boldsymbol{h}^{(j)} = (\underbrace{\alpha, \ldots, \alpha}_{k-j}, \underbrace{j\beta, \ldots, j\beta}_{n-k}, \underbrace{(j-1)\beta, \ldots, 2\beta, \beta, 0}_{j}), \tag{42}$$

and $h_{s_1} = \alpha$. To satisfy (38), we choose

$$\boldsymbol{h}' = (\underbrace{\alpha, \ldots, \alpha}_{s_1 - 1}, 0, \underbrace{\alpha, \ldots, \alpha}_{k-j-s_1}, \underbrace{(j+1)\beta, \ldots, (j+1)\beta}_{n-k}, \underbrace{j\beta, \ldots, 2\beta, \beta}_{j}), \tag{43}$$

which is a permutation of $\boldsymbol{h}^{(j+1)}$ and hence is in $\mathcal{H}$.

Case 3: If $m \leq j < m'$ and $s_1 \leq k - j$,

$$\boldsymbol{h} = \boldsymbol{h}^{(j)} = (\underbrace{\alpha, \ldots, \alpha}_{k-j}, \underbrace{j\beta, \ldots, j\beta}_{n-k}, \underbrace{(j-1)\beta, \ldots, 2\beta, \beta, 0}_{j}), \tag{44}$$

and $h_{s_1} = \alpha$. To satisfy (38), we choose

$$\boldsymbol{h}' = \boldsymbol{h}^{(j)} + (\underbrace{0, \ldots, 0}_{s_1 - 1}, -\alpha, \underbrace{0, \ldots, 0}_{k-m-s_1}, r, \underbrace{\beta, \ldots, \beta}_{n-k+m-1}), \tag{45}$$

which is a permutation of $\boldsymbol{h}^{(j+1)}$ and hence is in $\mathcal{H}$.

This establishes Property 5. ∎

Similar to the discussions at the end of Section III, the codes can be designed either via a randomized algorithm, or via a non-randomized algorithm based on Jaggi *et al.*'s algorithm. We can set up a virtual multicast problem, where each rank test $\boldsymbol{h}$ in $\mathcal{H}$ with $h_1 > 0$ corresponds to a receiver node.

V. Conclusion

In this paper we studied techniques for achieving the optimal tradeoff between storage efficiency and repair bandwidth, in a distributed storage system. Via a cut-based analysis, the optimal tradeoff between storage and repair bandwidth is characterized. Via an algebraic path-weaving technique, the existence of codes achieving the optimal tradeoff is established.

References

[1] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, "OceanStore: An Architecture for Global-scale Persistent Storage," in *Proc. ASPLOS*, Boston, MA, November 2000.

[2] A. G. Dimakis, P. G. Godfrey, M. J. Wainwright, and K. Ramchandran, "Network coding for peer-to-peer storage," in *IEEE Proc. INFOCOM*, Anchorage, Alaska, May 2007.

[3] S.-Y. R. Li, R. W. Yeung, and N. Cai, "Linear network coding," *IEEE Trans. on Information Theory*, vol. 49, pp. 371–381, February 2003.

[4] J. Bang-Jensen and G. Gutin, *Digraphs: Theory, Algorithms and Applications*. New York: Springer, 2001.

[5] R. Motwani and P. Raghavan, *Randomized Algorithms*. Cambridge University Press, 1995.

[6] S. Jaggi, P. Sanders, P. A. Chou, M. Effros, S. Egner, K. Jain, and L. Tolhuizen, "Polynomial time algorithms for network code construction," *IEEE Trans. Inform. Theory*, vol. 51, pp. 1973–1982, June 2005.

[7] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Trans. Networking*, vol. 11, no. 5, pp. 782–795, Oct. 2003.